Analysis of Community Smells Variability using ML Algorithms

Nidhi Kumari Chauhan
EECS, University of Ottawa
Ottawa, Canada
nidhi.chauhan@uottawa.ca

Mohana Prriya Sivakumar
EECS, University of Ottawa
Ottawa, Canada
msiva036@uotatwa.ca

***Abstract*—In software engineering, "social debt" is a colloquial term for unanticipated project costs associated with a "suboptimal" development community, wherein some of the characteristics noticed are challenges with coordination and communication. The presence of such inadequacies in the organizational systems can be determined by a marker, called Community smells. It is crucial to find and alleviate community smells, as they may well contribute to social debt. To address this, understanding the variability of the community smell, i.e., how various entities influence this factor to increase and decrease over time is needed. Additionally, the real predictive potential of such socio-technical measures is still unknown, though. In this study, we attempt to address this issue by conducting an empirical investigation by focusing on 4 community smells that are Organizational Silo, Black Cloud, Lone Wolf, and Radio Silence to (i) identify if there is there any correlation between 5 categories of socio-technical metrics—Developer Social Network Metrics, Socio-Technical Metrics, Core Community Members Metrics, Turnover, and Social Network Analysis Metrics and (ii) identify the possibility of predicting the variability of the community smell based on socio-technical code metrics. Our dataset, which is made up of 60 open-source applications, is used to achieve this goal. The key results of our study indicate the Jrip and Bagging with J48 as a base classifier performed the best with above 80% accuracy for Black cloud community smell. Whereas, Random forest performed the best for Radio Silence and Organizational Silo community smell. Finally, Jrip performed the best for the Lone Wolf Community Smell. This analysis will help the developers to reduce the community smells and therefore, improve the software quality which in-turn reduces the rework cost in the future.**

Keywords—Keywords: Community Smells, Social Debt, Software community

I. INTRODUCTION

Software engineering is a social activity [1,2] that frequently involves thousands, if not more, of stakeholders spread around the globe who are frequently geographically and culturally apart [3,4,5]. "Social debt" is a colloquial term for unanticipated project costs [6,7] associated with software development. The concept of social debt highlights the ways in which power dynamics and systemic inequalities within a community can result in some individuals or groups being expected to perform work without receiving the same level of recognition or compensation as others. Understanding and addressing social debt is important for the sustainability and health of software communities. This can create a sense of resentment and lead to a lack of diversity and inclusion within the community. To identify this, community smells is used as a marker.

Community smells, also known as social smells or social code smells, refer to patterns of behaviour in software development communities that can indicate a lack of collaboration, communication, or teamwork. These patterns may be related to the way that the community communicates, collaborates, or makes decisions, and they may be identified through observations or experiences of community members. These smells can have a negative impact on the quality, maintainability and evolution of software projects and [8, 9, 10] can lead to issues such as reduced productivity and increased conflict among team members. Examples of community smells may include:

- Lack of communication or collaboration between team members or groups.
- Inefficient processes or tools for managing work.
- Difficulty in decision-making.
- A lack of transparency or accountability in how work is carried out.
- Limited opportunities for learning or professional development.

The four different community smells considered in our study are Organizational Silo, Black Cloud, Lone Wolf, and Radio Silence.

Organizational Silo refers to a situation where different departments, teams, or functions within an organization operate independently of each other, with little communication or collaboration. This can lead to inefficiencies and conflicts within the organization, as each department or team may have its own goals and priorities that are not aligned with those of the organization as a whole.

The community smell Black Cloud can be caused by a variety of factors, such as a lack of leadership or clear direction, conflicts or disputes among members, or a lack of support or resources. It can also be impacted by negative behaviour or attitudes on the part of individual members, such as gossip, criticism, or a lack of respect for others. This can lead to discouragement of participation and engagement, which makes it difficult for members to feel motivated or valued. It can also create a sense of distrust and hostility among members, and make it difficult for the community to work effectively or achieve its goals.

A lone wolf is a term that is used to describe an individual who operates independently and does not seek or value the support or input of others. In a community or group setting, a lone wolf may be perceived as someone who is resistant to collaboration or teamwork, and who prefers to work alone or do things their own way. The presence of lone wolves within a community can be a source of conflict and tension, as they may not align with the values or goals of the group, and may not be willing to contribute to group efforts or initiatives. They may also be seen as difficult to work with or uncooperative, and may be perceived as not being fully committed to the success of the community.

The community smell, Radio silence is a situation where there is a lack of communication or engagement within a community or group. This can occur when members of the community are not actively participating in discussions or activities, or when there is a lack of response to messages or requests for input. It is caused due to a lack of interest or engagement on the part of community members, or a lack of clarity or purpose in the community's goals or activities. It can also be caused by external factors, such as busy schedules or competing demands on a member's time. It is a problem in a community because it can hinder the ability of the group to work effectively or achieve its goals. It can also lead to a sense of isolation or disconnection among members, and may make it difficult for the community to remain active or relevant.

Over the past few years, community smell research has drawn more interest. The relationship between community smells and their technical counterparts [11], as well as technical debt in general, has been demonstrated by recent research [12,13, 14].

One of the challenges in detecting and addressing community smells is the variability of these smells across different software development communities. Different communities may exhibit different patterns of behaviour and may have different cultural norms, which can affect the prevalence and impact of community smells.

Hence, to better understand and address community smells variability, researchers have proposed using machine learning techniques to analyse data from software development communities [15,16]. These techniques can help identify patterns and trends in community smells and can provide insights into the factors that contribute to their prevalence and impact. For this, the previously studied statistical approach on 40 socio-technical factors which contribute to the variability of four harmful community smells. They are Lone wolf, Radio Silence, Organizational Silo and Black Cloud, concerning 60 open-source software communities is taken as the base study.

The Key contribution in this paper is the proposal to analyse community smells variability using machine learning algorithms. Proceeding further to identify any possible correlation among the categories of the socio technical metrics; such as Developer Social Network metrics, Socio-Technical metrics, Core Community Members metrics, Turnover and Social Network Analysis Metrics. Our approach involves applying a range of machine learning techniques to a dataset of community smells data, and evaluating the effectiveness of these techniques in detecting and analysing community smells variability. Finally, discuss the implications of our results for understanding and addressing community smells in software development.

Structure of the paper. The rest of this paper is structured as follows. Section 2 outlines the background and related work. Sections 3 to 4 outlines our research design while Section 4 showcase our results. In Section 5 we discuss possible threats to validity of our results and explain how we mitigated them. Finally, we conclude the paper in Section 6.

II. Background and Related Work

A software development team's capacity for effective team building process may have an impact on the quality of the final product/ service. In this section, we discuss the literature related to community smells as well as the research conducted on other socio-technical aspects.

Traditionally, community smells have been detected and analyzed using manual inspection or specialized tools that rely on predefined rules or heuristics. However, these approaches can be time-consuming, error-prone, and limited in their ability to adapt to changing patterns and characteristics of community smells. Machine learning, on the other hand, has the potential to overcome these limitations by learning from data and predict community smells, which can help software teams address these issues and improve the overall health of the software community.

Since the advent of CODEFACE, there has been increased interest in community smells. The tool can establish so-called developer networks, or graphs depicting the actual relationships of collaboration and communication among developers, by mining code repositories and developer interactions. It then builds on these networks to automatically discover software communities. Naturally, the availability of this technology has facilitated additional field research.

Researchers have also been examining how these smells occur, what the key variables that may influence them are, and how developers actually refactor community smells. Catolino et al. [14] highlighted how increased gender diversity may be able to prevent the establishment of communal odours in some circumstances. An additional research looking at the perspectives of the practitioners [15] reported that developers believe that other characteristics, such as developer experience or team size, may make a community more prone to be smelly, however they do not view gender diversity and the inclusion of women in software teams as relevant criteria to avoid community smells. [15]

A large scale study was performed on 60 open-sourced software communities to evaluate the magnitude of community smell spread, how they are percieved and the relation to the existing socio-technical factors[13]. The study that served as the foundation for our paper also served as an additional piece of research for the paper mentioned earlier. It focused on analysing community smells to determine whether and to what extent socio-technical factors may affect the variability of the community smells, indicating an increase or decrease over time. The study's findings showed that each element taken into account gave some information gain for predicting community odours. Notably, the most effective predictors include indicators linked to turnover, communication, and socio-technical congruence. With the contributions made in this work, we want to offer a fundamental predictive mechanism with respectable accuracy figures.

III. IMPLEMENTATION

The implementation of our study is carried out with the help of Weka. It is a collection of machine learning algorithms and tools for data mining and data analysis. It is implemented in the Java programming language and is freely available under the GNU General Public License. Weka is widely used in academia and industry for research and development of machine learning models and techniques. It includes a graphical user interface (GUI) that allows users to interact with the algorithms and tools, as well as a suite of command-line tools for more advanced users.

The data set had been extracted, and section 4.1 describes how this was done. Additionally, data preprocessing is carried out to prepare the dataset for implementation. Cleaning is required, as well as the elimination of uncertainty,

management of missing values, and class-level mapping for the detection of code smells using static code metrics. This is done to clean up the dataset of any abnormalities and establish a connection between code smells and static code metrics. This process is thought to be crucial for maximizing the dataset's potential.

In order to choose features from the dataset and reduce the dimensionality of the data, which could affect the precision and accuracy of the results, feature selection approaches are now used. Feature selection is the process of selecting a subset of relevant features (also known as attributes or variables) from a larger set of features for use in building a machine learning model. The goal of feature selection is to identify the most relevant and useful features that will help improve the accuracy and performance of the model, while reducing the complexity and overfitting of the model.

Post feature selection process, it was identified that few features such as "*ml.only.devs*" had numerical values ranging from 98 to 820. Another feature, for an instance is "*mail.only.core.devs*" had numerical values ranging from 54 to 276. To handle this huge range of data and to normalize it, the feature scaling process is carried out. Feature scaling is the process of normalizing or standardizing the features (also known as attributes or variables) of a dataset so that they are on the same scale. This is important in machine learning because many algorithms use distance measures to evaluate the similarity or dissimilarity between data points. If the features are not on the same scale, some features may dominate others and may have a disproportionate influence on the results.

Feature Selection aims to improve the performance of the model by reducing the complexity and redundancy of the data, as well as to improve the interpretability of the model. Additionally, the following feature selector is used to complete the feature selection procedure, as shown below:

1. Information Gain

By determining the Information Gain in the manner described below, the Info Gain method assesses how closely an attribute is related to the class.

$$\text{InfoGain}(C, A) = H(C) - H(C|A)$$

They are assessed with the assistance of the data mining tool Weka5 (Version 3.9.4), and their use facilitates in putting the best static code metrics into practise, preserving both human and dataset processing time. The following section also includes the research questions and more implementation explanation.

IV. EMPIRICAL EVALUATION

In this section we will discuss the fundamental methodological overview, the approach followed to execute the implementation, and the respective results obtained.

4.1 Description of Dataset

The dataset for the project has been extracted from an online repository, GitHub. This dataset consists of numerical values of 40 distinct socio-technical metrics against 60 open-source software projects. The projects are carefully chosen based on 1) Codebase size, 2) Programming language, 3) Community size, and 4) Age of the active participants in the community. Additionally, the smell variability measures - "increase," "decrease," and "stable" - will serve as the target variable. For each of the four considered community smells, a data collection of 660 data points is supplied. The dataset obtained for this study was already labelled. Henceforth, the following variables will be used for the application of machine learning algorithm are as follows:

1. Target variable / Response variable: the target variable (also known as the dependent variable, response variable, or label) is the variable that is being attempted to be predicted or estimated. The selection of target variable is important as it will determine the type of model and the evaluation metric that is to be employed to measure the performance.

   We initially calculated the difference between the number of community smells in each time window pair (TWi, TWi+1) of a project: If the difference was greater than zero, it meant that there were more community smells from TWi to TWi+1, therefore it was labeled as "increase"; if it was less than zero, it meant there were fewer community smells and marked as "decrease"; otherwise, it was labeled as "steady".

2. Independent variable: a variable that is being manipulated or controlled in an experiment in order to observe the effect on the dependent variable. The independent variable for this project is the already mentioned 40 socio-technical factors.

4.2 Research Questions

Two primary research topics that aim to investigate the issue of predicting community smell variability serve as the foundation for our investigation. The knowledge regarding the quantity of socio-technical measures and the amount of community smells in the open-source projects has been identified as a concern from the prior study that served as the base paper. As a result, the following issues arose, which our research is intended to address. The following is a list of the questions.

RQ1: Is there any correlation between 5 categories of socio-technical metrics?

We intend to analyse the 40 socio-technical measures that are currently in use in this research subject. This analysis is the first step in the process to determine whether there is an opportunity to discover commonality between the metrics, and based on the findings, the next task is to further investigate to establish the prevalence between the socio-technical metrics, i.e., figuring out the correlation between the metrics that are divided into five different categories, namely categories—Developer Social Network Metrics, Socio-Technical Metrics, Core Community Members Metric, Turnover, and Social Network Analysis metrics.

To tackle this question the following implementation is carried out. The cleaned and processed dataset is then taken to compute the Pearson correlation to identify the sameness between the distinct set of categories. In machine learning, Pearson correlation is a measure of the linear correlation between two variables. It is a statistical technique that is used to determine the strength and direction of the relationship between two variables, which can be useful in understanding the relationships between different features in a dataset.

The Pearson correlation coefficient, also known as the Pearson's r, ranges from -1 to 1, where -1 indicates a strong negative correlation, 0 indicates no correlation, and 1 indicates a strong positive correlation. A value of 0 means that there is no relationship between the two variables, while a value of 1 or -1 indicates a strong relationship.

The Pearson correlation coefficient is calculated using the following formula:

$$r = \sum(x - \bar{x})(y - \bar{y})/(\sqrt{\sum(x - \bar{x})^2)(\sqrt{\sum(y - \bar{y})^2)}$$

where x and y are the two variables being measured, $\bar{x}$ and $\bar{y}$ are the means of the two variables, and Σ represents the sum of the values.

A positive correlation means that as the value of one variable increases, the value of the other variable also tends to increase. Whereas, a negative correlation means that as the value of one variable increases, the value of the other variable tends to decrease.

RQ2: Is it possible to predict the variability of the community smell based on socio-technical code metrics?

In this RQ, we tend to forecast the variability of community smell based on socio-technical metrics using machine learning techniques. This will undoubtedly help the developers to assist the variability of the community smells and further help to improve the resource usage and reduce overhead rework costs.

To handle this research question, the input variables from the available set of 40 socio-technical metrics columns is determined. All the useful columns from the extracted dataset is chosen as input variables, after excluding certain columns which were considered insignificant for the determination of variability of community smells. The target variable is then chosen as the output variable.

An observation sparked for the need of performing feature scaling on the obtained dataset. Feature scaling is a method used to standardize the range of independent variables or features of data. In machine learning, it is important to scale the features so that they are on the same scale, as features on different scales can weigh differently in the model. Following this, Weka is deployed to implement the machine learning algorithms.

Machine Learning Approach

Weka, a program that is free to use for data mining practise, is used to complete the machine learning exercise. Seven alternative machine learning algorithms are compared in order to achieve the detection rules, and they are as follows:

- JRip: It is an error-reducing propositional rule learner called Repeated Incremental Pruning (RIPPER).
- Random Forest: It operates by building a large number of decision trees during training period and then producing the class that represents the mean of the predictions (regression) or classifications of all the individual trees.
- J48: The decision tree algorithm is implemented by J48 in the Weka machine learning software suite. It is a classification algorithm that creates a decision tree model from a training dataset. J48 divides the data into smaller and smaller subsets using binary splits, resulting in data points that are pure, or include just one class, in each subset. Making predictions about the class of brand-new, unobserved data items is then possible using the resulting tree.
- Naïve Bayes: An estimator class for using the probabilities of certain features occurring in a given class to make predictions about the class of a new data point.
- MLP: A classifier that categorizes examples by learning a multi-layer perceptron using backpropagation. The network can be set up manually or with the use of a straightforward heuristic.
- Ibk(KNN): identifies the k data points in the training set that are most similar to a given test point, and using the class labels or values of those points to make a prediction about the class.
- AdaBoost with J48 as classifier: Adaboost is an ensemble learning algorithm that is used to improve the performance of the base classifier J48 decision trees.
- Bagging: This method implements training of multiple models on different random subsets of the training data and combining their predictions to make a final prediction.

The table 1 depicts the various 40 socio-technical metrics involved in the study along with their description.

4.3 Analysis Procedure and Metrics

There are currently 40 socio-technical metrics available, which are grouped into five categories. We will analyze all the socio-technical metrics and will aim to find the correlations. Further, feature selection techniques was applied based on community smells. Additionally, to anticipate the community smells variability, the pre-processed dataset was used and Multi-Class Classification algorithms such as J48, Naïve Bayes, MLP, Ibk(KNN), Adaboost with J48 classifier, and Bagging.

The following performance metrics are deployed for performance evaluation,

- Precision: The number of accurate positive predictions (TP) divided by the total number of positive predictions (TP + FP) is used to calculate precision.
- Recall: It is derived by dividing the total number of positive predictions by the proportion of correct positive predictions (TP) (P).
- F-measure: It is defined as the harmonic mean of precision and recall.

  F1=2*(precision*recall)/(precision+recall)
- ROC Area: a graph showing the trade-off between the true positive rate (TPR) and the false positive rate (FPR).
- MCC: It is the measure of the quality of a binary classification model.

Table 1: 40 Socio-Technical Metrics

Category	Metric	Description
Developer Social Network metrics	devs	Number of developers present in the global Developers Social Network
	ml.only.devs	Number of developers present only in the communication Developers Social Network
	code.only.devs	Number of developers present only in the collaboration Developers Social Network
	ml.code.devs	Number of developers present both in the collaboration and in the communication DSNs
	perc.ml.only.devs	Percentage of developers present only in the communication Developers Social Network
	perc.code.only.devs	Percentage of developers present only in the collaboration Developers Social Network
	perc.ml.code.devs	Percentage of developers present both in the collaboration and in the communication DSNs
	sponsored.devs	Number of sponsored developers (95% of their commits are done in working hours)
	ratio.sponsored	Ratio of sponsored developers with respect to developers present in the collaboration DSN
Socio-Technical Metrics	st.congruence	Estimation of socio-technical congruence
	communicability	Estimation of information communicability (decisions diffusion)
	num.tz	Number of timezones involved in the software development
	ratio.smelly.devs	Ratio of developers involved in at least one Community Smell
Core community members metrics	core.global.devs	Number of core developers of the global Developers Social Network
	core.mail.devs	Number of core developers of the communication Developers Social Network
	core.code.devs	Number of core developers of the collaboration Developers Social Network
	sponsored.core.devs	Number of core sponsored developers
	ratio.sponsored.core	Ratio of core sponsored developers with respect to core developers of the collaboration DSN
	global.truck	Ratio of non-core developers of the global Developers Social Network
	mail.truck	Ratio of non-core developers of the communication Developers Social Network
	code.truck	Ratio of non-core developers of the collaboration Developers Social Network
	mail.only.core.devs	Number of core developers present only in the communication DSN
	code.only.core.devs	Number of core developers present only in the collaboration DSN
	ml.code.core.devs	Number of core developers present both in the communication and in the collaboration DSNs
	ratio.mail.only.core	Ratio of core developers present only in the communication DSN
	ratio.code.only.core	Ratio of core developers present only in the collaboration DSN
	ratio.ml.code.core	Ratio of core developers present both in the communication and in the collaboration DSNs
Turnover	global.turnover	Global developers turnover with respect to the previous temporal window
	code.turnover	Collaboration developers turnover with respect to the previous temporal window
	core.global.turnover	Core global developers turnover with respect to the previous temporal window
	core.mail.turnover	Core communication developers turnover with respect to the previous temporal window
	core.code.turnover	Core collaboration developers turnover with respect to the previous temporal window
	ratio.smelly.quitters	Ratio of developers previously involved in any Community Smell that left the community
Social Network Analysis metrics	closeness.centr	SNA degree metric of the global DSN computed using closeness
	betweenness.centr	SNA degree metric of the global DSN computed using betweenness
	degree.centr	SNA degree metric of the global DSN computed using degree
	global.mod	SNA modularity metric of the global DSN
	mail.mod	SNA modularity metric of the communication Developers Social Network
	code.mod	SNA modularity metric of the collaboration Developers Social Network
	density	SNA density metric of the global Developers Social Network

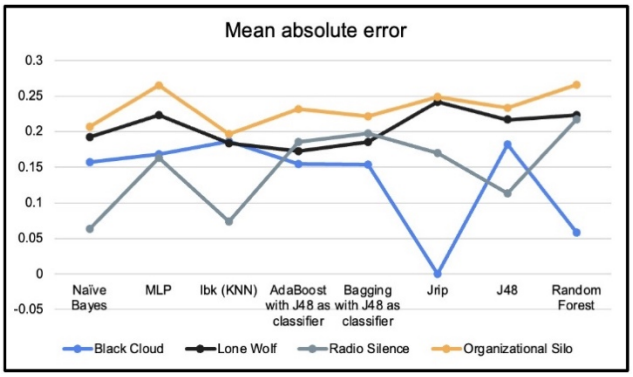

Figure1: Mean Absolute Error for Community Smells

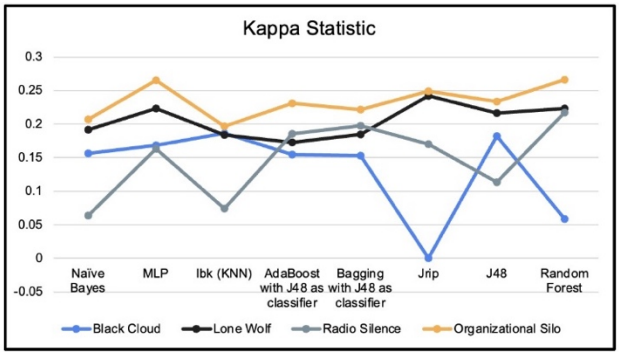

Figure 2: Kappa Statistic for Community Smells

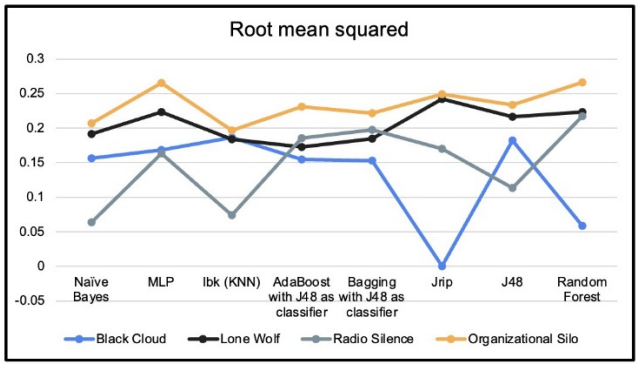

Figure 1: Root mean squared metric for Community Smells

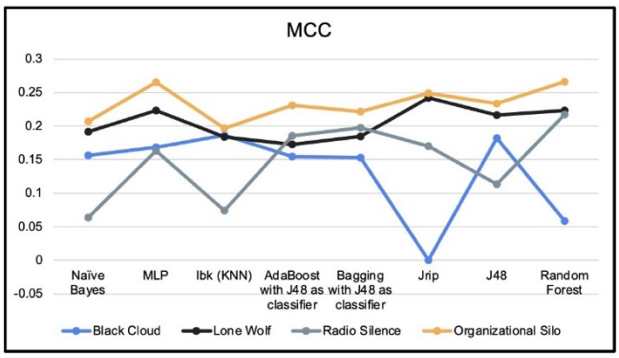

Figure 3: MCC Metric for Community Smells

The figure 1 provides with the insights of variability in four community smells according to the performance metric: Mean Absolute Error. The figure 2 provides with the insights of variability in four community smells according to the performance metric: Root mean squared. The figure 3 provides with the insights of variability in four community smells according to the performance metric: Kappa Statistic. The figure 4 provides with the insights of variability in four community smells according to the performance metric: MCC respectively.

From observing the performance metrices, it is observed that two community smells namely, Lone wolf and Organizational Silo shows less variation in results across all the machine learning algorithms. With respect to the Radio Silence community smell, It performed well to a lesser extent compared to the two previous community smells. Significant drop in results is noticed with respect to the J48 algorithm and Random Forest Performed the best with this community smell.

Another significant observation is seen with respect to the Black cloud community smell. This is one community smell in which significant difference in values ie., greater variance in values is seen. This performance significantly fell with the usage of the algorithms Jrip and Random Forest.

4.4 Results and Discussion

RQ1: The Pearson correlation was obtained between the 40 socio-technical metrics and the following insights were observed.

- Metric like 'dev', 'ml.only.devs' and 'code.only.devs' which belongs to Category 'Developer Social Network Metric' have high correlation as seen from the Pearson-Correlation graph (0.79 between devs and ml.only.devs,, 0.74 between devs and code.only.devs)
- Category 'Socio-Technical Metrics' had very less correlation between them as seen in 0.62 correlation between 'st.congruence' and 'communicability', -0.0024 between 'st.congruence' and 'num.tz' and -0.16 between 'st.congruence' and 'ratio.smelly.devs'
- In Category 'Core-community member metrics' have variable correlation 0.90 between 'core.global.devs' and 'core.mail.devs', 0.701 between 'core.global.devs' and 'core.code.devs' and low correlation like '-0.069 between 'core.global.devs' and 'code.truck'.
- In category 'Turnover', the metrics are much in correlation with each other, 0.43 between 'global.turnover' and 'code.turnover', 0.38 between 'global.turnover' and 'core.global.turnover'. 0.38 between 'global.turnover' and 'core.mail.turnover'.
- 'Social Network Analysis metrics' category has very less correlation as seen 0.38 between 'closeness.centr' and 'betweenness.centr', 0.37 between 'closeness.centr' and 'degree centr', -0.28 between 'closeness.centr' and 'global.mod', -0.32 between 'closeness.centr' and 'mail.mod'.

RQ2: The four community smells were manipulated against 8 different Machine learning approaches and the findings from the experiment is depicted as following,

- For the “Black Cloud” Community smell, the Bagging algorithm with the base classifier J48 and Jrip algorithm performed very well and obtained the highest accuracy of 82.5758% as shown in Table 2.
- For the “Lone Wolf” Community smell, the Jrip algorithm performed the best among other algorithms, obtaining 53.0303% as illustrated in Table 3.
- With respect to the “Radio Silence” Community smell, the MLP algorithm performed well with the accuracy rate of 57.8788% as depicted in Table 5.
- Finally, for the community smell “Organizational Silo”, both the algorithms; MLP, and Random forest performed well with same 53.6364% accuracy as shown in Table 4.

4.5 Threats to validity

This section describes potential risks that might have impacted our findings and how we handled them.

Construct Validity. The accuracy of the dataset used in the study is a concern for threats in this category. The independent and response variables we included in our models were already available and computed for all the time frames taken into account in the study since we drew on a publically accessible source that had been constructed in the context of earlier research. The results show that all of the community smells produced by the tool are true positives, and that extra smell instances were not called out by developers, hence lowering the possibility of false negatives. We are assured of the response variable's reliability as a result of this validation.

Conclusion Validity. The statistical techniques used pose a significant danger to the conclusions reached. The fact that our response variable was categorical and had three possible values led us to use the Multinomial Logistic Linear statistical technique. The statistical approach is suitable for the problem at hand since it can handle multiclass problems with categorical and continuous independent variables.

External validity. Our study takes into account 60 open-source communities that are all active but varied in terms of size, breadth, number of contributors, and domain in order to assess the generalizability of the findings. We are aware, however, that the performance of the models under test may vary on other datasets. As a result, we welcome replications of our research.

V. CONCLUSION

In this paper, we sought to understand the variability of community smells over time, considering 60 software projects and statistically analyzing how 40 socio-technical metrics relate to the increase/decrease of four community smell types. Our findings revealed the factors that community shepherds should monitor to avoid the proliferation of specific community-related issues.

Our future research agenda reflects our aim to employ different set of Community smells, Machine learning algorithms, and ensembles which will provide the in-depth understanding of the community smells on a large scale to reduce rework and maintenance costs in the software communities.

Table 2: Performance Measure for Black Cloud

BC	Accuracy	Precision	Recall	F-Measure	ROC Area
Naïve Bayes	55.7576	0.802	0.558	0.632	0.75
MLP	76.9697	0.752	0.77	0.761	0.761
Ibk (KNN)	75.9091	0.759	0.759	0.759	0.614
AdaBoost with J48 as classifier	80.6061	0.752	0.806	0.773	0.745
Bagging with J48 as classifier	**82.5758**	0.764	0.826	0.78	0.786
Jrip	**82.5758**	0.69	0.826	0.752	0.495
J48	78.1818	0.754	0.782	0.767	0.696
Random Forest	82.4242	0.75	0.824	0.764	0.807

Table 3: Performance Measure for Lone Wolf

Lone Wolf	Accuracy	Precision	Recall	F-Measure	ROC Area
Naïve Bayes	44.8485	0.479	0.448	0.41	0.648
MLP	51.8182	0.516	0.518	0.517	0.633
Ibk (KNN)	49.5455	0.494	0.495	0.495	0.579
AdaBoost with J48 as classifier	49.0909	0.489	0.491	0.489	0.61
Bagging	49.8485	0.498	0.498	0.497	0.632
Jrip	**53.0303**	0.53	0.53	0.529	0.646
J48	50.7576	0.494	0.508	0.484	0.604
Random Forest	52.1212	0.521	0.521	0.521	0.649

Table 4: Performance Measure for Radio Silence

Radio Silence	Accuracy	Precision	Recall	F-Measure	ROC Area
Naïve Bayes	31.5152	0.496	0.315	0.349	0.557
MLP	54.8485	0.525	0.548	0.535	0.63
Ibk (KNN)	48.1818	0.482	0.482	0.482	0.538
AdaBoost with J48 as classifier	55.1515	0.544	0.552	0.547	0.629
Bagging with J48 as classifier	56.6667	0.552	0.567	0.556	0.637
Jrip	55.6061	0.549	0.556	0.541	0.588
J48	51.3636	0.505	0.514	0.508	0.563
Random Forest	**57.8788**	0.559	0.579	0.563	0.657

Table 5: : Performance Measure for Organizational Silo

Organizational Silo	Accuracy	Precision	Recall	F-Measure	ROC Area
Naïve Bayes	44.8485	0.456	0.448	0.394	0.668
MLP	**53.6364**	0.541	0.536	0.537	0.674
Ibk (KNN)	49.2424	0.494	0.492	0.493	0.588
AdaBoost with J48 as classifier	51.5152	0.519	0.515	0.516	0.636
Bagging with J48 as classifier	50.7576	0.51	0.508	0.509	0.659
Jrip	52.5758	0.53	0.526	0.528	0.649
J48	50.7576	0.504	0.508	0.504	0.617
Random Forest	**53.6364**	0.54	0.536	0.538	0.685

REFERENCES

[1] Paul Ralph, Mike Chiasson, and Helen Kelley. Social theory for software engineering research. In Sarah Beecham, Barbara A. Kitchenham, and Stephen G. MacDonell, editors, EASE, pages 44:1–44:11. ACM, 2016.

[2] Anna Hannemann, Hans-Jorg Happel, Matthias Jarke, Ralf Klamma, Steffen Lohmann, Walid Maalej, and Volker Wulf. Social aspects in software engineering. In Software Engineering (Workshops), volume 150 of LNI, pages 239–242. GI, 2009.

[3] Helen Sharp, Hugh Robinson, and Mark Woodman. Software engineering: Community and culture. IEEE Software, 17(1):40–47, 2000.

[4] Lucas Gren. On gender, ethnicity, and culture in empirical software engineering research. CoRR, abs/1904.09820, 2019.

[5] Hannu Jaakkola. Culture sensitive aspects in software engineering. In Antje Dusterhoft, Meike Klettke, and Klaus-Dieter Schewe, editors, Conceptual Modelling and Its Theoretical Foundations, volume 7260 of Lecture Notes in Computer Science, pages 291–315. Springer, 2012.

[6] "On the nature of social debt in software development communities" by Rachel Kobayashi and Matthew J.K. T. Lem (published in the proceedings of the 39th International Conference on Software Engineering in 2017).

[7] "Exploring the Role of Social Debt in Open Source Software Development" by Rachel Kobayashi and Matthew J.K. T. Lem (published in the IEEE Transactions on Software Engineering in 2019).

[8] F. Palomba, D. A. A. Tamburri, F. Arcelli Fontana, R. Oliveto, A. Zaidman, and A. Serebrenik. Beyond technical aspects: How do community smells influence the intensity of code smells? IEEE Transactions on Software Engineering, pages 1–1, 2018.

[9] D. A. Tamburri. Software architecture social debt: Managing the incommunicability factor. IEEE Transactions on Computational Social Systems, 6(1):20–37, Feb 2019.

[10] D. A. Tamburri, F. Palomba, and R. Kazman. Exploring community smells in open-source: An automated approach. IEEE Transactions on Software Engineering, pages 1–1, 2019.

[11] A. Martini and J. Bosch, “Revealing social debt with the CAFFEA framework: An antidote to architectural debt,” in 2017 IEEE International Conference on Software Architecture Workshops. IEEE Computer Society, 2017, pp. 179–181.

[12] A. Martini, T. Besker, and J. Bosch, “Technical debt tracking: Current state of practice: A survey and multiple case study in 15 large organizations,” Science of Computer Programming, vol. 163, pp. 42–61, 2018.

[13] Jianxiong Dong and Qiangfu Zhao, "An Empirical Study of Community Smell in Open Source Software Projects" in the 39th International Conference on Software Engineering in 2017.

[14] "Predicting and Mitigating Community Smells in Open Source Software Projects: A Machine Learning Approach" by Saurabh Mishra, Pradeep K. Chaturvedi, and Manish Kumar. IEEE Transactions on Software Engineering in 2021.

[15] "Community Smell Detection Using Machine Learning: A Review and Synthesis" by Saurabh Mishra, Pradeep K. Chaturvedi, and Manish Kumar. IEEE Access, 2020.

[16] M. Joblin, W. Mauerer, S. Apel, J. Siegmund, and D. Riehle, “From developer networks to verified communities: A fine-grained approach,” in Proceedings of the 37th International Conference on Software